



US Patent & Trademark Office

[Subscribe \(Full Service\)](#) [Register \(Limited Service, Free\)](#) [Login](#)

Search: ☒ The ACM Digital Library ☐ The Guide

asynchronous software testing + sequence + procedure call



THE ACM DIGITAL LIBRARY



[Feedback](#) [Report a problem](#) [Satisfaction survey](#)

Terms used

asynchronous software testing sequence procedure call

Found 87,107 of 151,219

Sort results
by

relevance



[Save results to a Binder](#)

[Try an Advanced Search](#)

[Try this search in The ACM Guide](#)

Display
results

expanded form



[Search Tips](#)

☐ Open results in a new window

Results 1 - 20 of 200

Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

Best 200 shown

Relevance scale ☐ ☐ ☐ ☐ ☐

1 [Human-computer interface development: concepts and systems for its management](#)

H. Rex Hartson, Deborah Hix

March 1989 **ACM Computing Surveys (CSUR)**, Volume 21 Issue 1

Full text available: [pdf\(7.97 MB\)](#)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)

Human-computer interface management, from a computer science viewpoint, focuses on the process of developing quality human-computer interfaces, including their representation, design, implementation, execution, evaluation, and maintenance. This survey presents important concepts of interface management: dialogue independence, structural modeling, representation, interactive tools, rapid prototyping, development methodologies, and control structures. *Dialogue independence* is th ...

2 [Distributed systems - programming and management: On remote procedure call](#)

Patricia Gomes Soares

November 1992 **Proceedings of the 1992 conference of the Centre for Advanced Studies on Collaborative research - Volume 2**

Full text available: [pdf\(4.52 MB\)](#)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#)

The Remote Procedure Call (RPC) paradigm is reviewed. The concept is described, along with the backbone structure of the mechanisms that support it. An overview of works in supporting these mechanisms is discussed. Extensions to the paradigm that have been proposed to enlarge its suitability, are studied. The main contributions of this paper are a standard view and classification of RPC mechanisms according to different perspectives, and a snapshot of the paradigm in use today and of goals for t ...

3 [Fast detection of communication patterns in distributed executions](#)

Thomas Kunz, Michiel F. H. Seuren

November 1997 **Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborative research**

Full text available: [pdf\(4.21 MB\)](#)


Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Understanding distributed applications is a tedious and difficult task. Visualizations based on process-time diagrams are often used to obtain a better understanding of the execution of the application. The visualization tool we use is Poet, an event tracer developed at the University of Waterloo. However, these diagrams are often very complex and do not provide the user with the desired overview of the application. In our experience, such tools display repeated occurrences of non-trivial commun ...

4 Protocol testing: review of methods and relevance for software testing

Gregor V. Bochmann, Alexandre Petrenko

August 1994 **Proceedings of the 1994 ACM SIGSOFT international symposium on Software testing and analysis**

Full text available:  pdf(2.22 MB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)

Communication protocols are the rules that govern the communication between the different components within a distributed computer system. Since protocols are implemented in software and/or hardware, the question arises whether the existing hardware and software testing methods would be adequate for the testing of communication protocols. The purpose of this paper is to explain in which way the problem of testing protocol implementations is different from the usual problem of software testi ...

5 Gate-level test generation for sequential circuits

Kwang-Ting Cheng

October 1996 **ACM Transactions on Design Automation of Electronic Systems (TODAES)**, Volume 1 Issue 4

Full text available:  pdf(448.19 KB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

This paper discusses the gate-level automatic test pattern generation (ATPG) methods and techniques for sequential circuits. The basic concepts, examples, advantages, and limitations of representative methods are reviewed in detail. The relationship between gate-level sequential circuit ATPG and the partial scan design is also discussed.

Keywords: IC testing, automatic test generation, sequential circuit test generation, testing

6 A survey of structured and object-oriented software specification methods and techniques

Roel Wieringa

December 1998 **ACM Computing Surveys (CSUR)**, Volume 30 Issue 4

Full text available:  pdf(605.26 KB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)

This article surveys techniques used in structured and object-oriented software specification methods. The techniques are classified as techniques for the specification of external interaction and internal decomposition. The external specification techniques are further subdivided into techniques for the specification of functions, behavior, and communication. After surveying the techniques, we summarize the way they are used in structured and object-oriented methods and indicate ways in w ...

Keywords: languages

7 A survey of asynchronous remote procedure calls

A. L. Ananda, B. H. Tay, E. K. Koh

April 1992 **ACM SIGOPS Operating Systems Review**, Volume 26 Issue 2

Full text available:  pdf(910.89 KB)

Additional Information: [full citation](#), [abstract](#), [citations](#), [index terms](#)

Remote Procedure Call (RPC) is a popular paradigm for interprocess communication in distributed systems. It is simple, flexible and powerful. However, most of the RPC systems today are synchronous in nature, and hence fail to exploit fully the parallelism inherent in distributed applications. In view of this, various asynchronous RPC systems have been designed and implemented to achieve higher parallelism while retaining the familiarity and


simplicity of synchronous RPC. Asynchronous RPC calls d ...

Keywords: asynchronous RPC, distributed systems, high-throughput, interprocess communication (IPC), intra-machine call, low-latency, parallelism, remote procedure call (RPC), synchronous RPC, transport-independent

8 APPL/A: a language for software process programming

Stanley M. Sutton, Dennis Heimbigner, Leon J. Osterweil

July 1995 **ACM Transactions on Software Engineering and Methodology (TOSEM)**,
Volume 4 Issue 3

Full text available:  [pdf\(4.89 MB\)](#)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)

Software process programming is the coding of software processes in executable programming languages. Process programming offers many potential benefits, but their realization has been hampered by a lack of experience in the design and use of process programming languages. APPL/A is a prototype software process programming language developed to help gain this experience. It is intended for the coding of programs to represent and support software processes including process, product, and p ...

Keywords: consistency management, multiparadigm programming languages, software process programming, transaction management

9 System-level power optimization: techniques and tools

Luca Benini, Giovanni de Micheli

April 2000 **ACM Transactions on Design Automation of Electronic Systems (TODAES)**,
Volume 5 Issue 2

Full text available:  [pdf\(385.22 KB\)](#)


Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

This tutorial surveys design methods for energy-efficient system-level design. We consider electronic systems consisting of a hardware platform and software layers. We consider the three major constituents of hardware that consume energy, namely computation, communication, and storage units, and we review methods of reducing their energy consumption. We also study models for analyzing the energy cost of software, and methods for energy-efficient software design and compilation. This survey ...

10 A Survey of Some Theoretical Aspects of Multiprocessing

J. L. Baer

January 1973 **ACM Computing Surveys (CSUR)**, Volume 5 Issue 1


Full text available:  [pdf\(4.05 MB\)](#)

Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

11 Computing curricula 2001

September 2001 **Journal on Educational Resources in Computing (JERIC)**

Full text available:  [pdf\(613.63 KB\)](#)

 [html\(2.78 KB\)](#)

Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

12 A software instruction counter

J. M. Mellor-Crummey, T. J. LeBlanc


April 1989 **ACM SIGARCH Computer Architecture News**, Proceedings of the third international conference on Architectural support for programming

languages and operating systems, Volume 17 Issue 2Full text available:  pdf(997.70 KB)Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Although several recent papers have proposed architectural support for program debugging and profiling, most processors do not yet provide even basic facilities, such as an instruction counter. As a result, system developers have been forced to invent software solutions. This paper describes our implementation of a software instruction counter for program debugging. We show that an instruction counter can be reasonably implemented in software, often with less than 10% execution overhead. Ou ...

13 On randomization in sequential and distributed algorithms

Rajiv Gupta, Scott A. Smolka, Shaji Bhaskar

March 1994 **ACM Computing Surveys (CSUR)**, Volume 26 Issue 1Full text available:  pdf(8.01 MB)Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Probabilistic, or randomized, algorithms are fast becoming as commonplace as conventional deterministic algorithms. This survey presents five techniques that have been widely used in the design of randomized algorithms. These techniques are illustrated using 12 randomized algorithms—both sequential and distributed—that span a wide range of applications, including: primality testing (a classical problem in number theory), interactive probabilistic proofs ...

Keywords: Byzantine agreement, CSP, analysis of algorithms, computational complexity, dining philosophers problem, distributed algorithms, graph isomorphism, hashing, interactive probabilistic proof systems, leader election, message routing, nearest-neighbors problem, perfect hashing, primality testing, probabilistic techniques, randomized or probabilistic algorithms, randomized quicksort, sequential algorithms, transitive tournaments, universal hashing

14 Tools for building asynchronous servers to support speech and audio applications

Barry Arons

December 1992 **Proceedings of the 5th annual ACM symposium on User interface software and technology**Full text available:  pdf(946.22 KB)Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Distributed client/server models are becoming increasingly prevalent in multimedia systems and advanced user interface design. A multimedia application, for example, may play and record audio, use speech recognition input, and use a window system for graphical I/O. The software architecture of such a system can be simplified if the application communicates to multiple servers (e.g., audio servers, recognition servers) that each manage different types of input and output. This paper describes ...

Keywords: asynchronous message passing, audio servers, distributed client-server architecture, remote procedure call, speech and studio applications, speech recognition and synthesis

15 Guide for the use of the Ada Ravenscar Profile in high integrity systems


Alan Burns, Brian Dobbing, Tullio Vardanega

June 2004 **ACM SIGAda Ada Letters**, Volume XXIV Issue 2Full text available:  pdf(548.17 KB)Additional Information: [full citation](#), [references](#)

16 The family of concurrent logic programming languages

Ehud Shapiro

September 1989 **ACM Computing Surveys (CSUR)**, Volume 21 Issue 3

Full text available:  [pdf\(9.62 MB\)](#)


Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Concurrent logic languages are high-level programming languages for parallel and distributed systems that offer a wide range of both known and novel concurrent programming techniques. Being logic programming languages, they preserve many advantages of the abstract logic programming model, including the logical reading of programs and computations, the convenience of representing data structures with logical terms and manipulating them using unification, and the amenability to metaprogrammin ...

17 Concepts and Notations for Concurrent Programming

Gregory R. Andrews, Fred B. Schneider

January 1983 **ACM Computing Surveys (CSUR)**, Volume 15 Issue 1

Full text available:  [pdf\(4.02 MB\)](#)

Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

18 Tutorial: Compiling concurrent languages for sequential processors

Stephen A. Edwards

April 2003 **ACM Transactions on Design Automation of Electronic Systems (TODAES)**, Volume 8 Issue 2

Full text available:  [pdf\(771.65 KB\)](#)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)


Embedded systems often include a traditional processor capable of executing sequential code, but both control and data-dominated tasks are often more naturally expressed using one of the many domain-specific concurrent specification languages. This article surveys a variety of techniques for translating these concurrent specifications into sequential code. The techniques address compiling a wide variety of languages, ranging from dataflow to Petri nets. Each uses a different method, to some degree ...

Keywords: Compilation, Esterel, Lustre, Petri nets, Verilog, code generation, communication, concurrency, dataflow, discrete-event, partial evaluation, sequential

19 Constrained expressions: toward broad applicability of analysis methods for distributed software systems

Laura K. Dillon, George S. Avrunin, Jack C. Wileden

July 1988 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 10 Issue 3

Full text available:  [pdf\(2.30 MB\)](#)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)

It is extremely difficult to characterize the possible behaviors of a distributed software system through informal reasoning. Developers of distributed systems require tools that support formal reasoning about properties of the behaviors of their systems. These tools should be applicable to designs and other preimplementation descriptions of a system, as well as to completed programs. Furthermore, they should not limit a developer's choice of development languages. In this paper ...

20 Programming languages for distributed computing systems

Henri E. Bal, Jennifer G. Steiner, Andrew S. Tanenbaum

September 1989 **ACM Computing Surveys (CSUR)**, Volume 21 Issue 3

Full text available:  [pdf\(6.50 MB\)](#)Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)

When distributed systems first appeared, they were programmed in traditional sequential languages, usually with the addition of a few library procedures for sending and receiving messages. As distributed applications became more commonplace and more sophisticated, this ad hoc approach became less satisfactory. Researchers all over the world began designing new programming languages specifically for implementing distributed applications. These languages and their history, their underlying pr ...

Results 1 - 20 of 200

Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2005 ACM, Inc.

[Terms of Usage](#) [Privacy Policy](#) [Code of Ethics](#) [Contact Us](#)Useful downloads:  [Adobe Acrobat](#)  [QuickTime](#)  [Windows Media Player](#)  [Real Player](#)